

HARDWARE-ACCELERATED HIGH AVAILABILITY INTEGRATED
NETWORKED STORAGE SYSTEM

Roger Thorpe et al

Appl. No.: 10/781,338 Atty Docket: ISTOR.013A



1/38

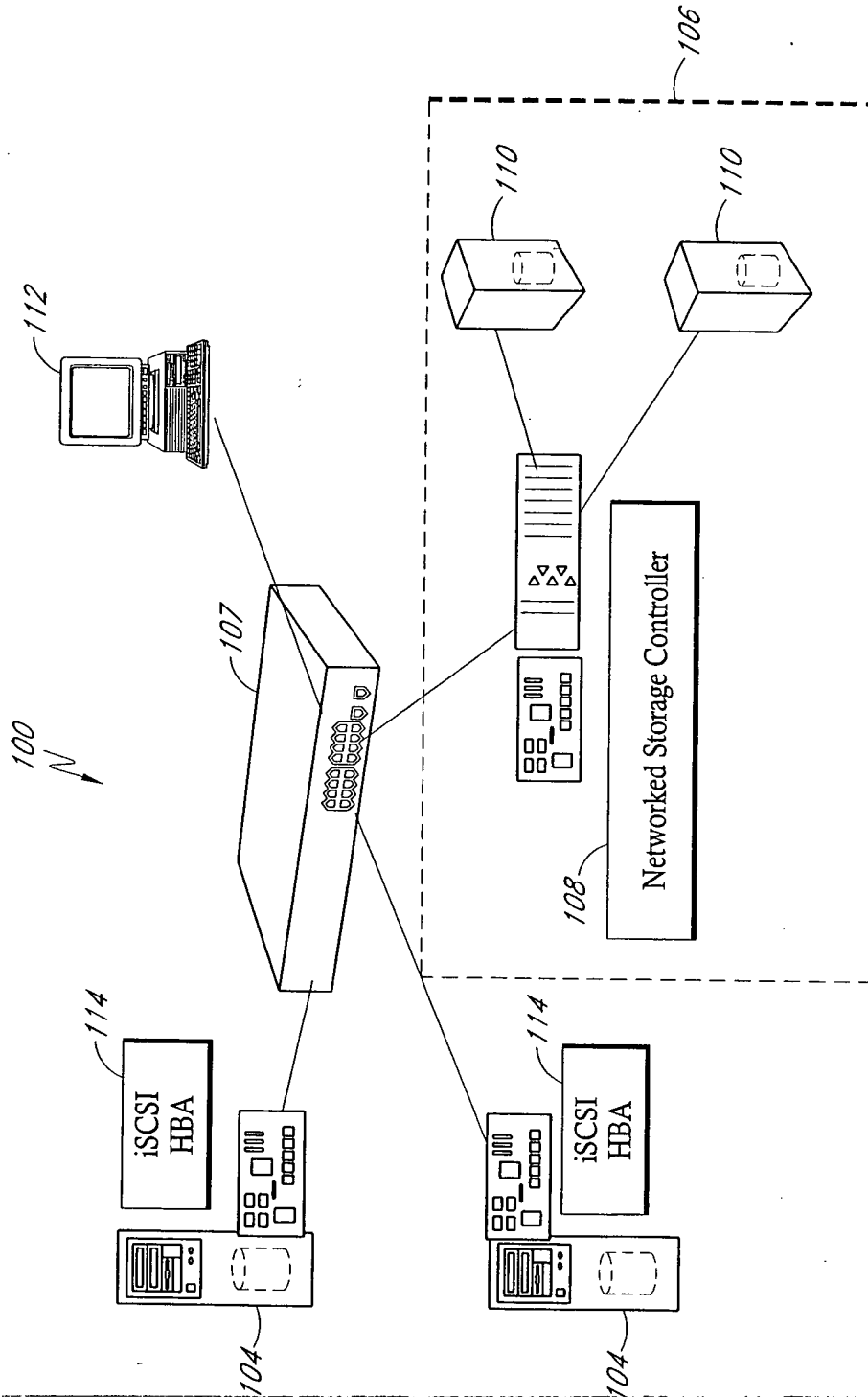


FIG. 1

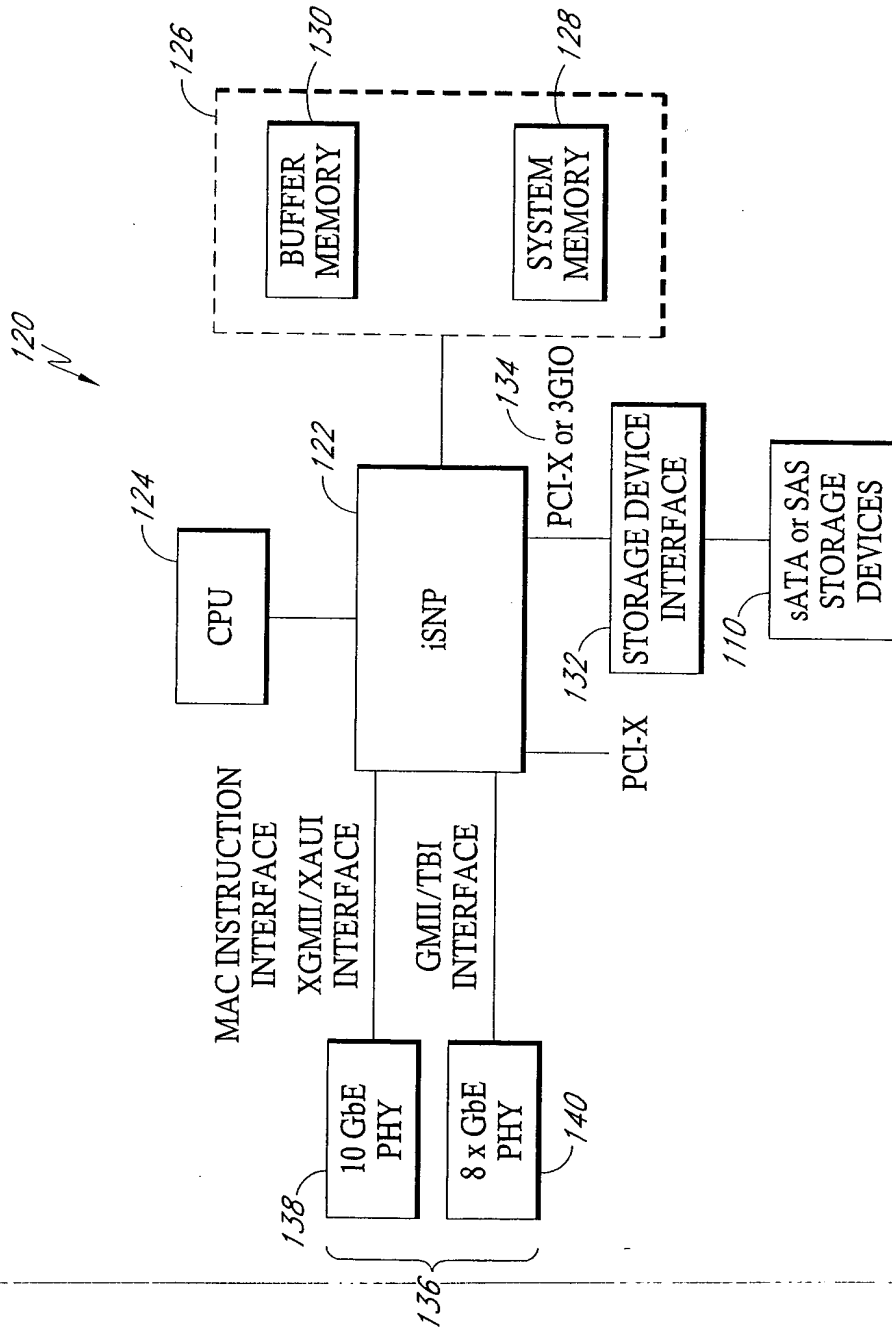
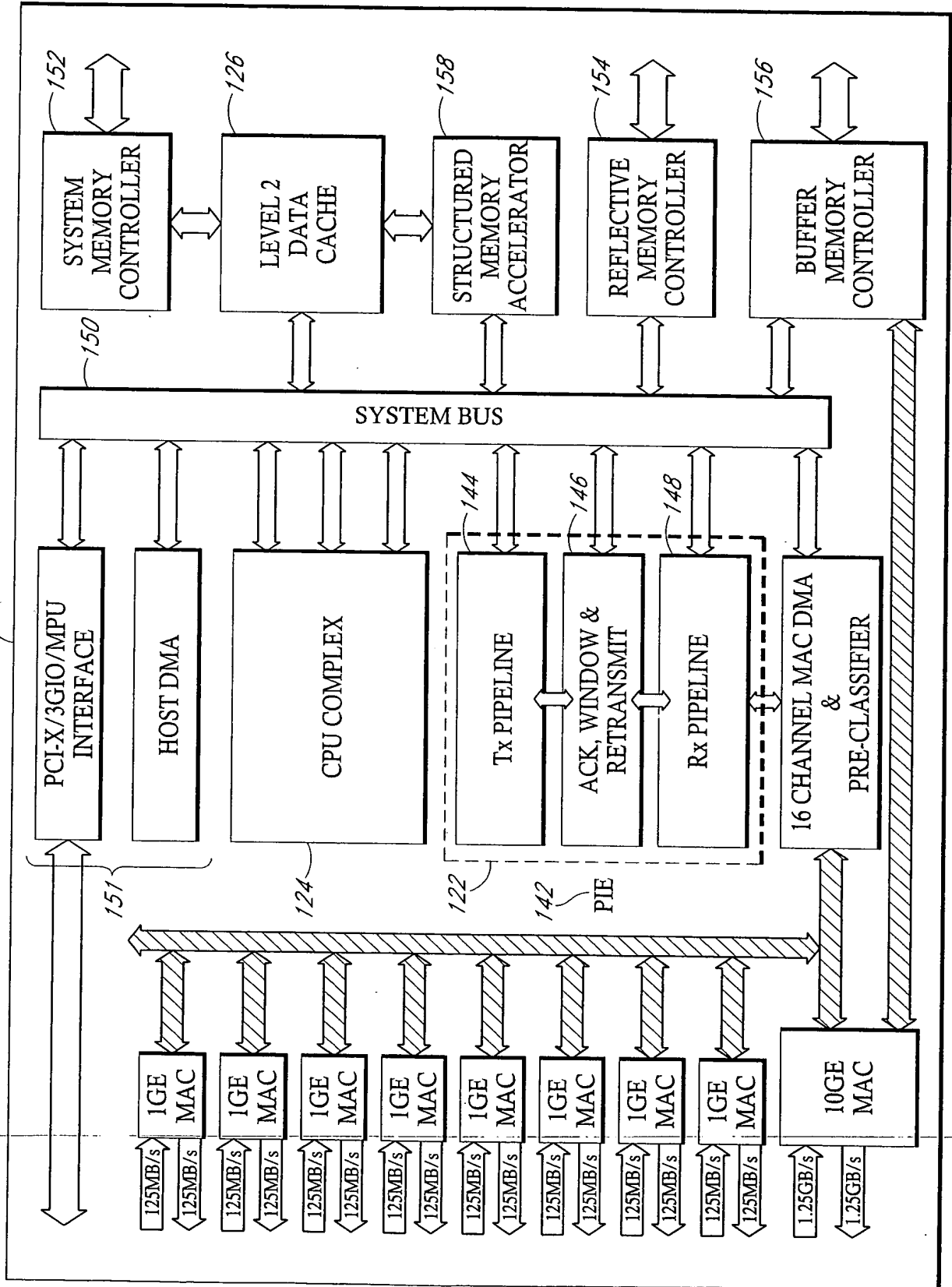


FIG. 2

3/38

FIG. 3A



4/38

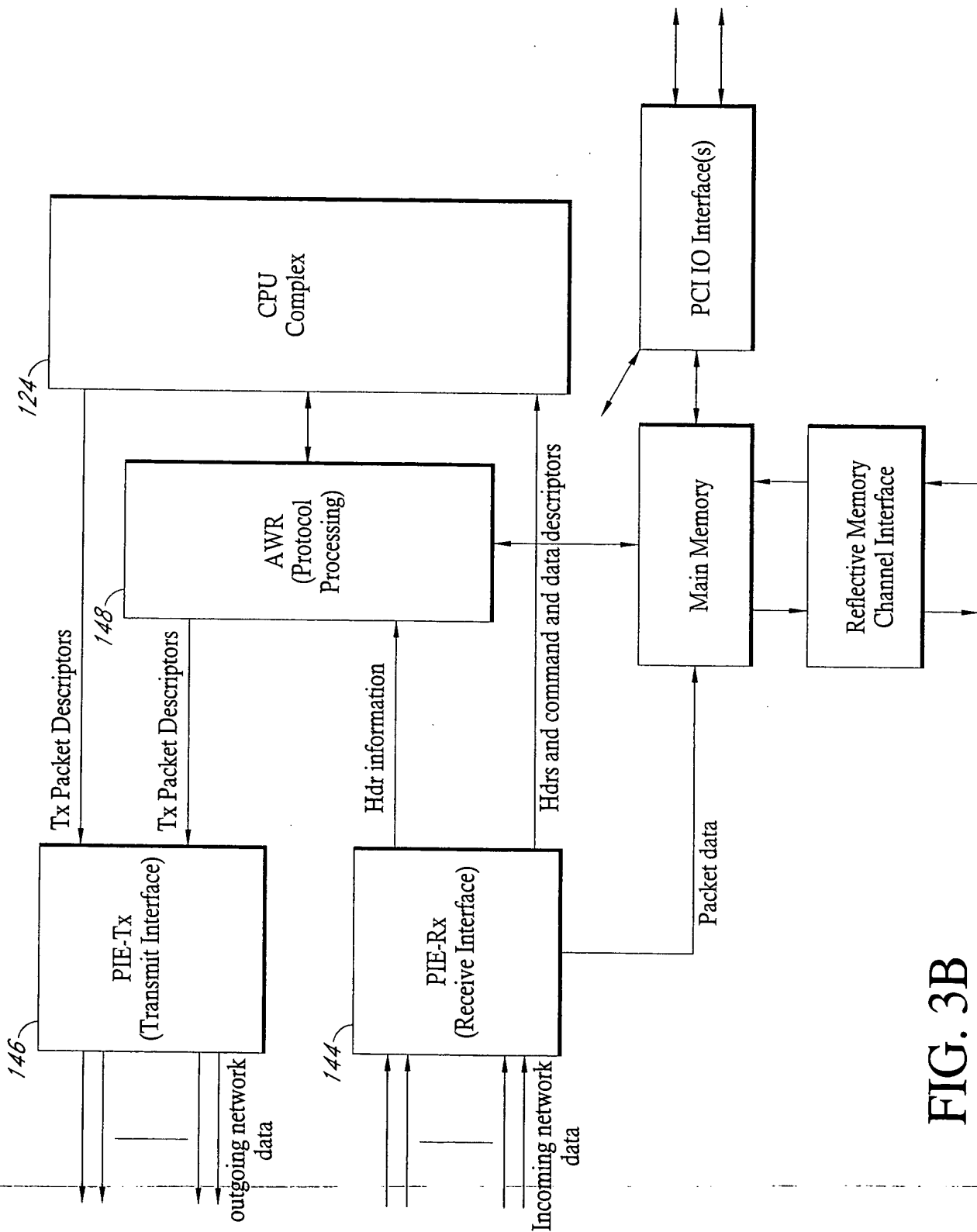


FIG. 3B

5/38

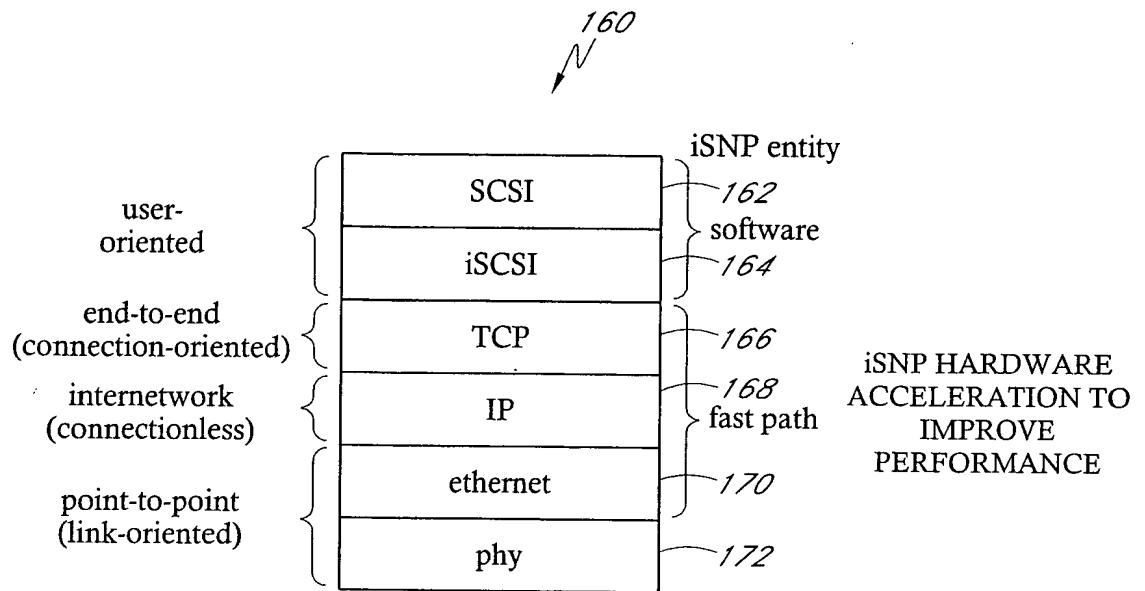


FIG. 4A

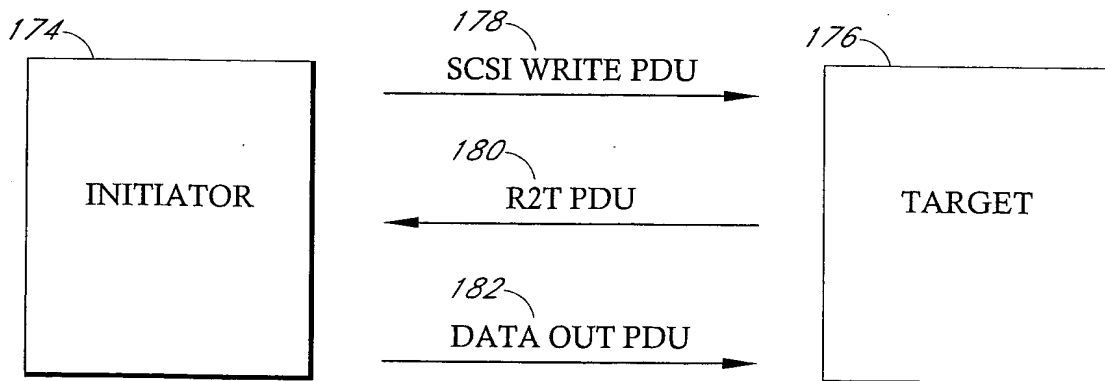


FIG. 4B

segment	iSCSI PDU	
BHS (basic header segment, bytes 0-47)	B0[6]=I (immed delivery) B0[5:0]=opcode, initiator h01=SCSI command h02=SCSI task mgmt req h03=login command h05=SCSI data out h06=logout command h10=SNACK (seq# ack) req h31'=reject	B1-B3-opcode specific fields SCSI command: B1[7]=F (final), B1[6]=R (read), B1[5]=W (write) B1[2:0]=task attrib (0=untagged, 1=simple, 2=ord, 3=hoq, 4=aca) B3=CRN (command reference number) SCSI data out: B1[7]=F SCSI data in: B1[7]=F, B1[2]=O (overflow), B1[1]=U (underflow), B1[0]=S (status, if set: B3=status, B20-23=residual count, B24-27=statusSN) SCSI response: B1[2]=O, B1[1]=U, B1[4]=O for bidir read, B1[3]=U bi rd B2=response (0=cmd complete, 1=targ fail)
	B4=total AHS length	B5-7=data segment length (not including pad)
	B8-15=logical unit number (LUN) (64 bits)	
	B16-19=initiator task tag	
	B20-47=opcode specific fields (7 words)	
	SCSI command: B20-23=expected data length, B24-27=cmdSN (cmd seq number), B28-31=expStatSN (acks status thru SN-1)	
	R2T: B20-23=targ transfer tag, B24-27=statSN (for next status), B28-31=expCmdSN (cmd ack to init), B32-35=maxCmdSN	
	B36-39=R2TSN (R2T PDU number, starts at 0), B40-43=buffer offset, B44-47=desired data length (in bytes)	
	SCSI data out: B20-23=targ transfer tag from R2T (or F's), B28-31=expStatSN (expected status sequence number)	
	B36-39=dataSN, B40-43=buffer offset (for this payload relative to complete data transfer)	
ABS (additional header seg. optional)	SCSI data in: B20-23=residual count, B24-27=statSN, B28-31=expCmdSN, B36-39=expDataSN,	
	SCSI response: B24-27=statSN, B28-31=expCmdSN, B36-39=expDataSN,	
	B40-43=bidirectional read residual count, B44-47=residual count	
	B0=B1=AHS length	B2=AHS type B2[7]=drop B2[5:0]=ahs code 1=extended CDB 2=exp bidir read length B3=AHS specific
	extended CDB: B4..n=extended CDB (if necessary pad to full word)	
hdr digest (optional)	bidir read length: B4=expected read data length	
data seg. (optional)	CRC for header segment(s)	
data digest (optional)	Data (if necessary pad to full word)	
	CRC for data segment	

FIG. 5A

7/38

TCP Segment Header Fields	
word#	
0	source port[15:0]
1	destination port[15:0]
2	sequence number[31:0]
3	acknowledgment number [31:0] (seq# of next expected octet, acks all previous octets)
4	reserved[5:0] flags[5:0]
5	hdr length[3:0] (in words)
6	bit5=urgent ptr valid bit4=ack number valid bit3=PSH bit2=RST (reset connection) bit1=SYN (for connect close) bit0=FIN (for close)
7	checksum[15:0] (same as IP, but covers header and data)
8	urgent pointer[15:0] (number of octets the receiver is able to receive)
9	options (if any) plus padding (variable length depending on the number and type of options)
10	data

FIG. 5B

8/38

IP Packet Header Fields				
word#	version[3:0]	IHL[3:0] (header len in words, min 5=no options)	TOS[7:0] (type of service, specifics precedence, delay, throughput and reliability parameters)	TLEN[15:0] (total length including header, in octets)
0				
1	ID[15:0] (with src addr, dst addr, and user protocol uniquely identifies the IP datagram)			flag[2:0] flag {1}=dont frag (in 64-bit units) flag[0]=more frags
2	TTL[7:0] (time to live, decremented at each hop, if 0 discard the datagram)	Protocol[7:0] 1=ICMP 6=TCP 17=UDP		Header checksum[15:0] (1's complement of the 1's complement sum of data interpreted 16 bits at a time, with end-around carry)
3	Source IP address[31:0] (coded to allow a variable number of bits to specify the network and station)			
4	Destination IP address[31:0] (as for source address, 224-239 .x.x.x=multicast)			
:	Options (if any) plus padding (variable length depending on the number and type of options)			
:	Data (multiple of 8 bits in length)			

FIG. 5C

ethernet frame field	pre- amble	SFD (start frame)	DA (dst addr)	SA (src addr)	type (ethernet II) or length of info (IEEE 802.3)	information	FCS (frame chk seq)	extension
#octets	7	1	6	6	2	46-1500	4	
notes					'h0800=IP 'h0806=ARP 'h86dd=IPv6 'h8100=VLAN (insert w/2-byte tag before type/length) <='h05dc=length (rfc 1042, insert w/constant 0xaaa03_000000 before type)	jumbo: max 9000 GE: min=512 for half duplex CSMA/CD) <min: add pad bytes	4	special nondata symbols for half duplex CSMA/CD

FIG. 5D

9/38

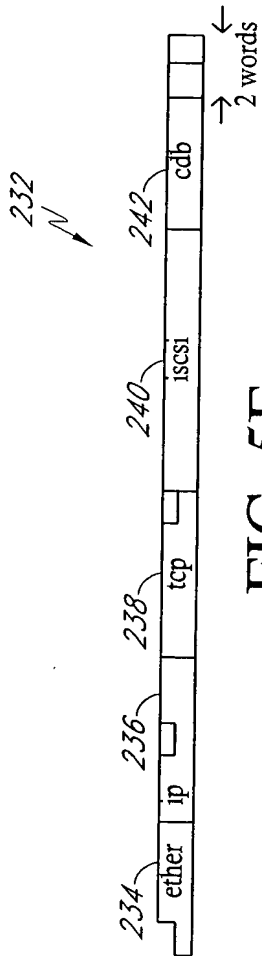


FIG. 5E

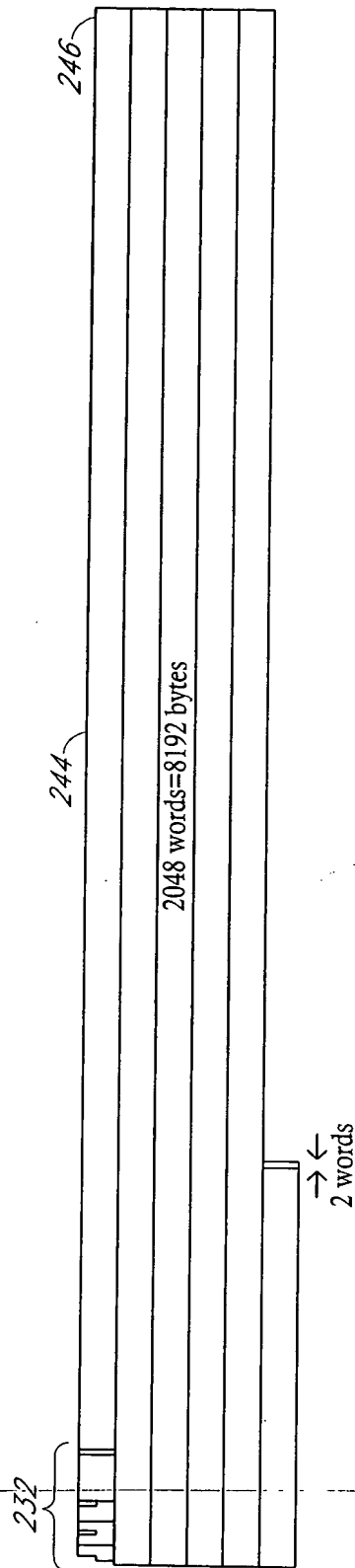


FIG. 5F

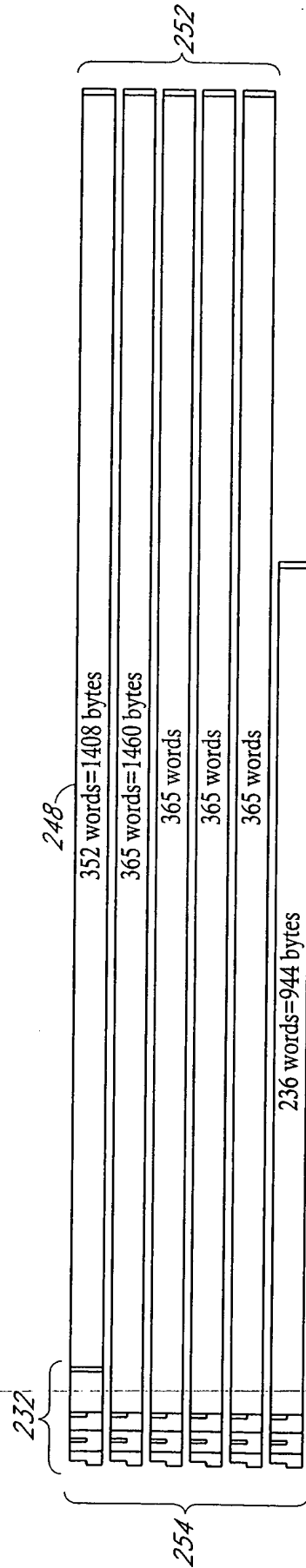


FIG. 5G

10/38

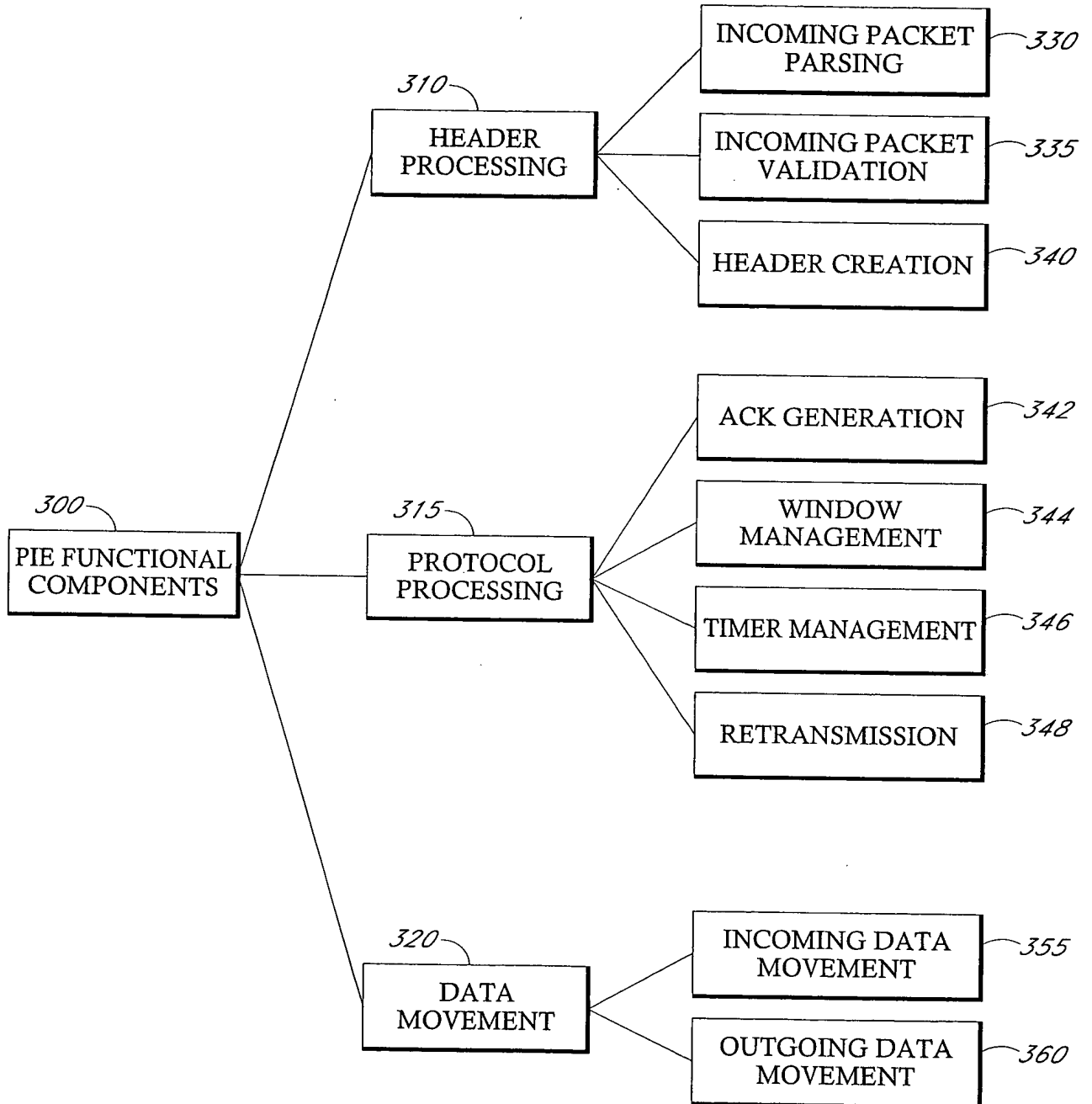


FIG. 6A

11/38

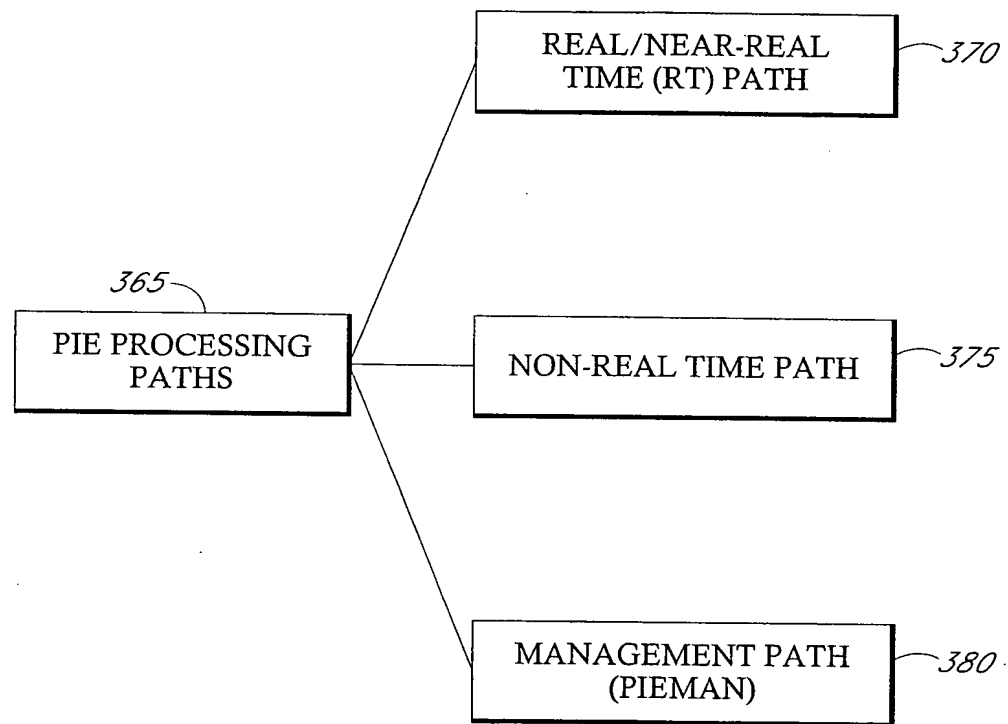


FIG. 6B

12/38

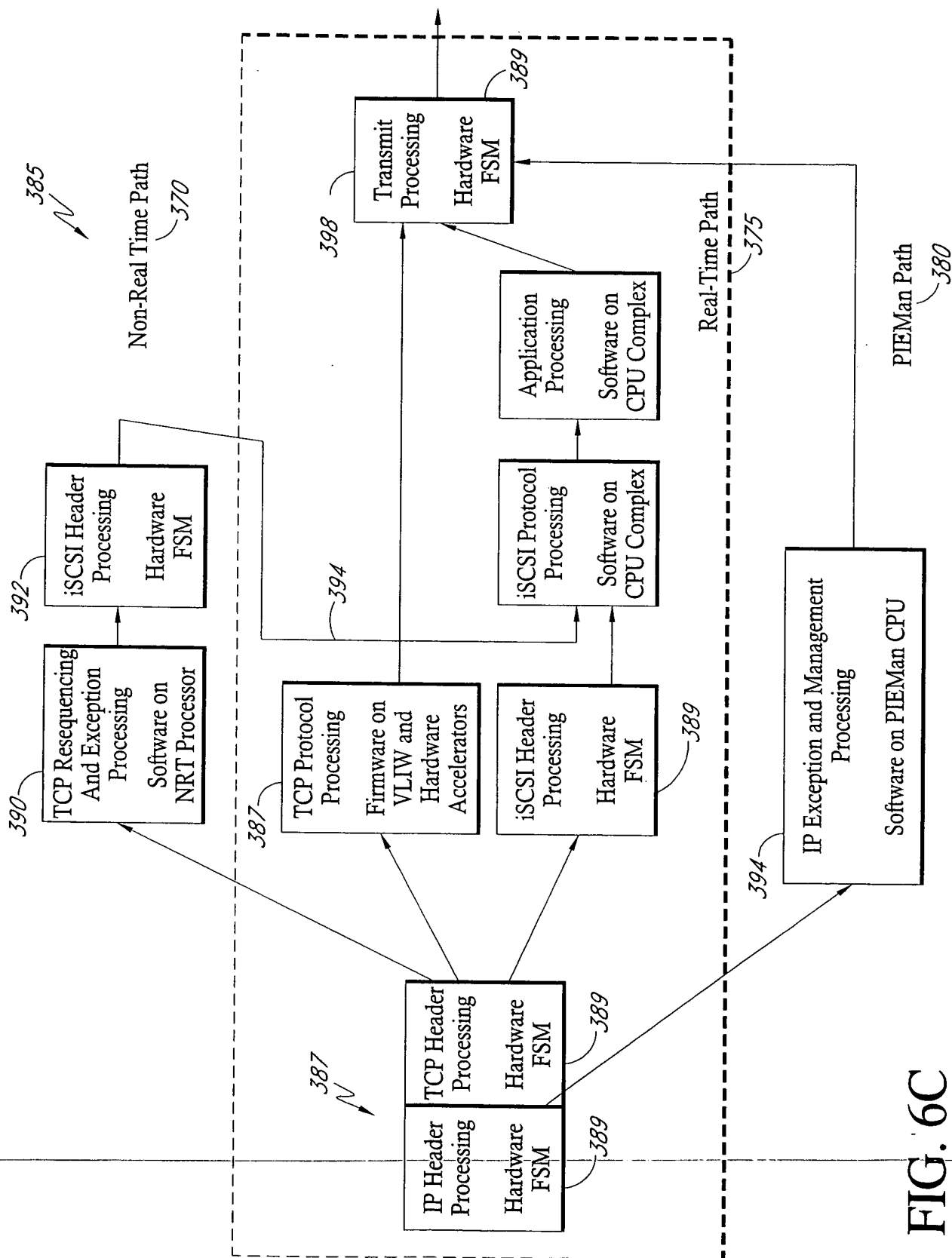
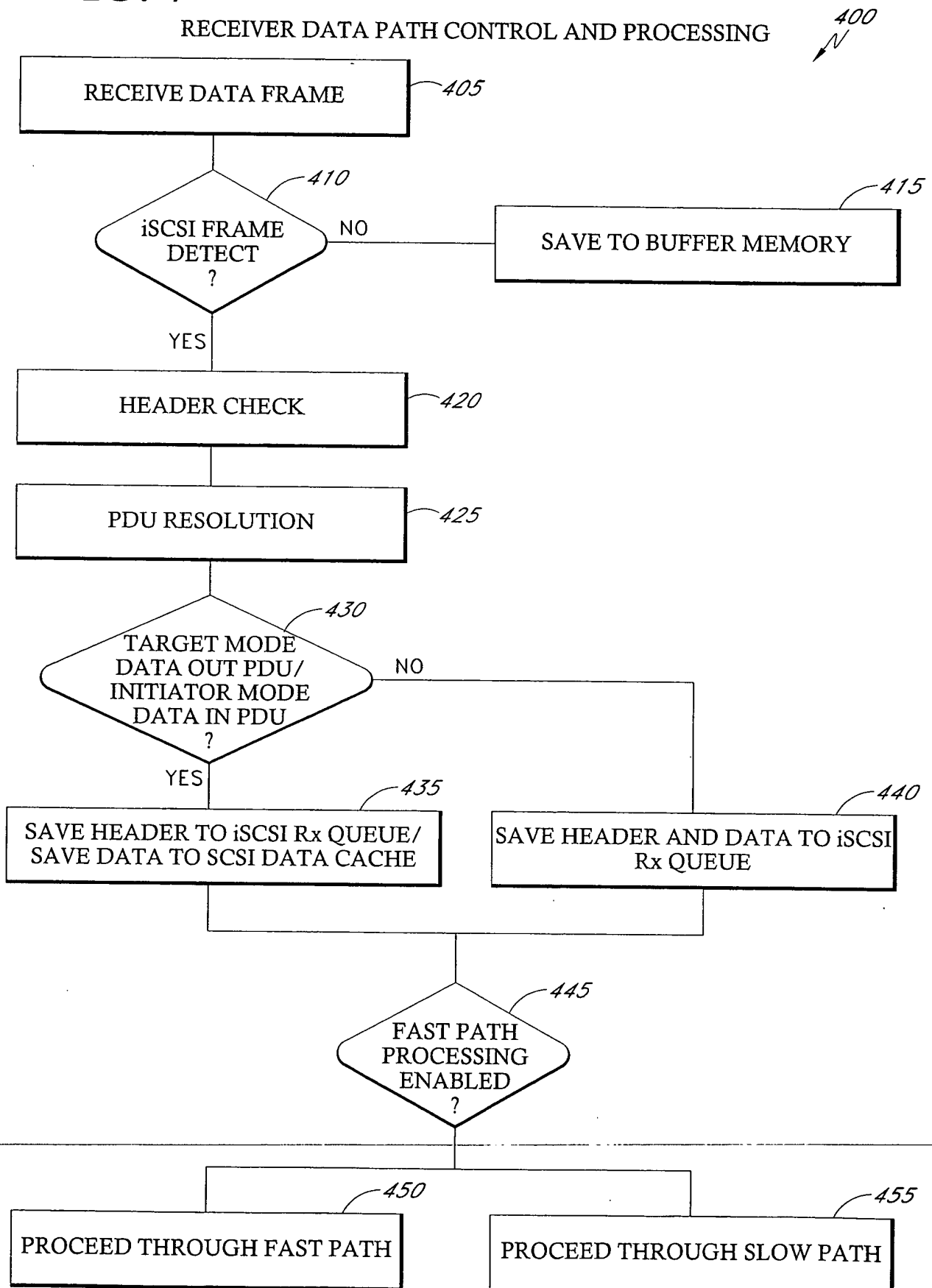


FIG. 6C

FIG. 7

13/38

RECEIVER DATA PATH CONTROL AND PROCESSING



14/38

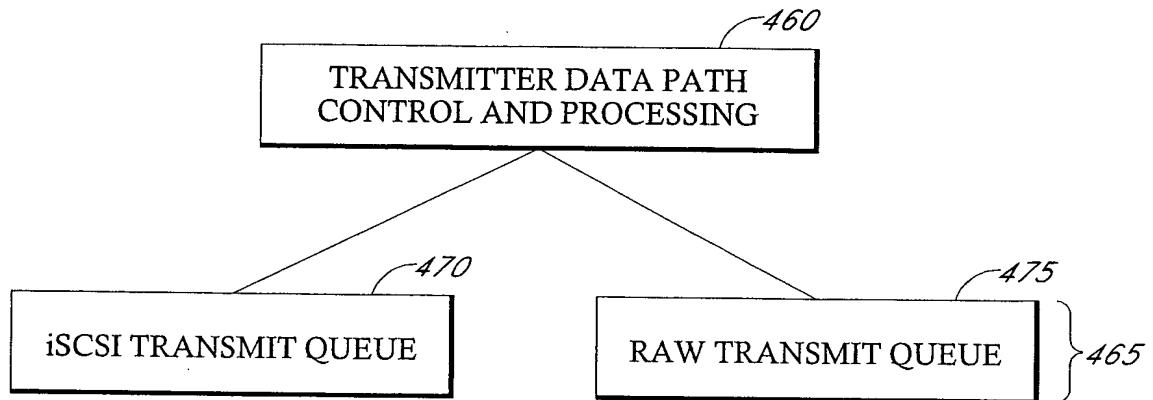


FIG. 8A

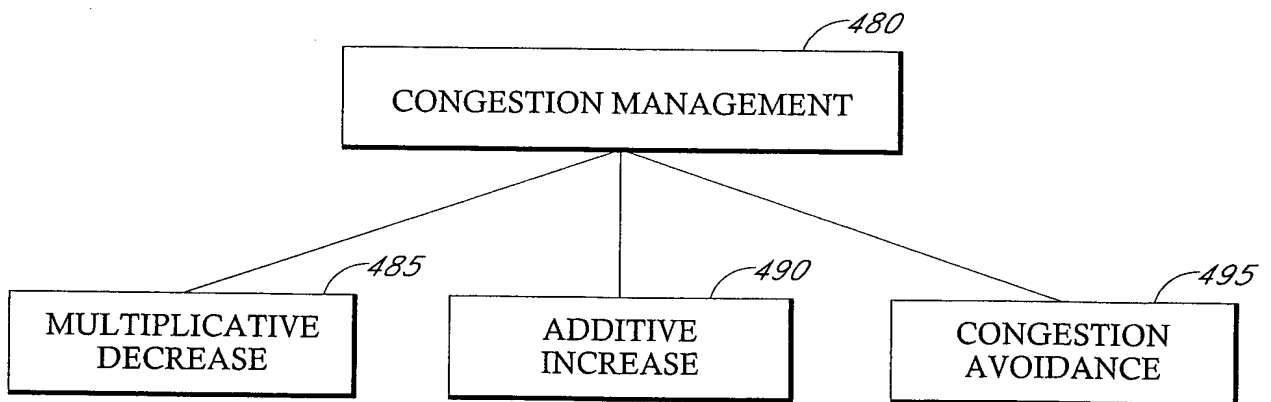


FIG. 8B

15/38

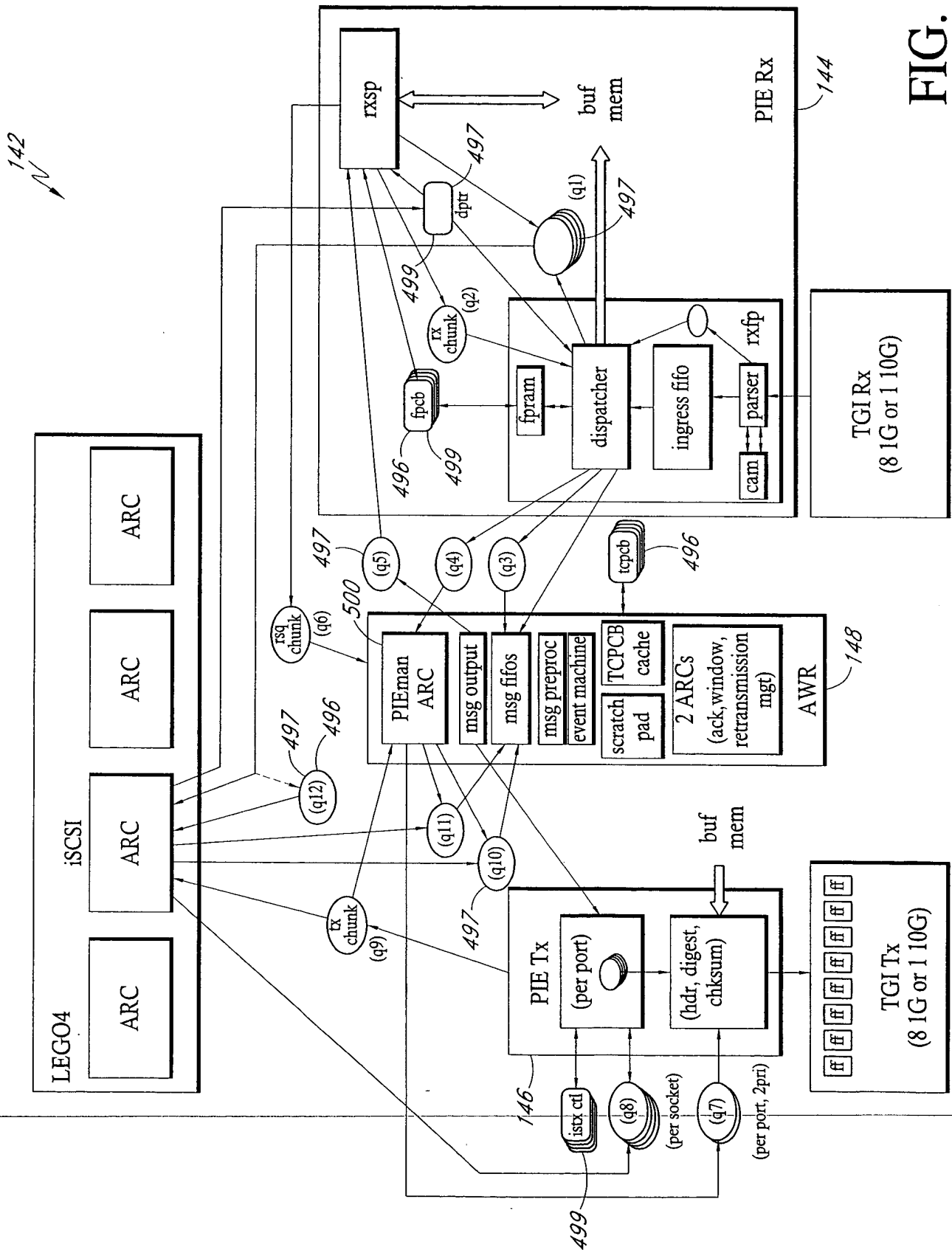


FIG. 9

16/38

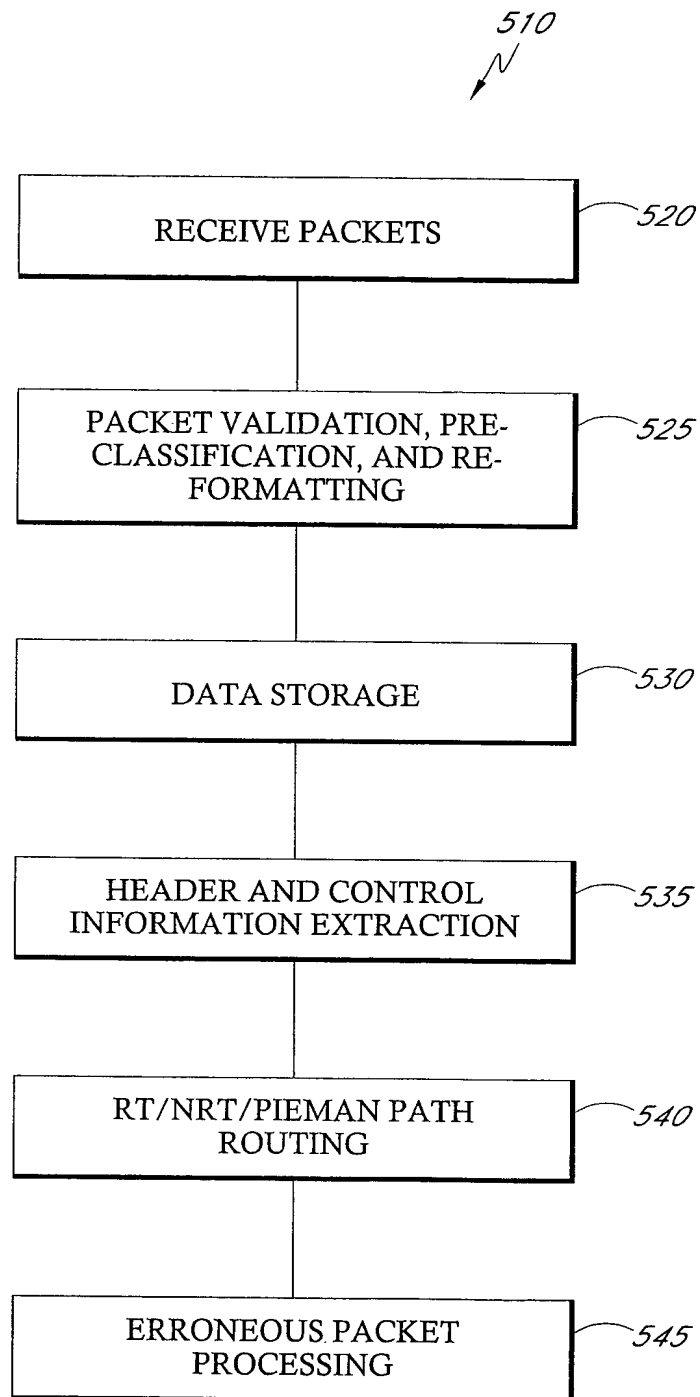


FIG. 10



HARDWARE-ACCELERATED HIGH AVAILABILITY INTEGRATED
NETWORKED STORAGE SYSTEM

Roger Thorpe et al

Appl. No.: 10/781,338 Atty Docket: ISTOR.013A

18/38

TGI field	Description
tag[3:0]	<p>Indicates the type of information in the dword, as preclassified by the TGI</p> <p>0: invalid (interframe)</p> <p>1: E (ethernet header)</p> <p>2: EV (ethernet header, VLAN present)</p> <p>3: E8 (ethernet header, 802.3 rfc1042 format)</p> <p>4: EV8 (ethernet header, 802.3 rfc1042 format, VLAN present)</p> <p>5: I (IP)</p> <p>6: IO (IP options)</p> <p>7: IF (IP fragmented)</p>
off[2:0]	<p>8: T (TCP)</p> <p>9: TO (TCP option)</p> <p>10: U (UDP)</p> <p>11: spare</p> <p>12: S (iSCSI, i.e. TCP and DPORT or SPORT matches iSCSI)</p> <p>13: o (other, e.g. not IP, not TCP or UDP)</p> <p>14: G (good EOF, dword holds checksum, frame length)</p> <p>15: B (bad EOF, assert early if checksum or other error detected)</p> <p>Indicates the byte offset to tag boundary within dword, 0=left side</p>

FIG. 12A

B0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
raw non-vlan and vlan frames															
tag,off															
e 0															
i 0															
t 4															
t 8															
ev 0															
i 0															
t 4															
t 8															
ev8 0															
i 0															
t 4															
t 8															
e8 0															
i 0															
t 4															
t 8															

#bits	Field	Description
4	State	State machine state
2	ts_offset	Timestamp offset (0=none, 1=22B from start of TCP header, 2=23B, 3=24B)
1	t_left	TCP header starts in left half of dword
4	Reason	If nonzero, the reason from slow-path processing
1	Msw_parity	Job FIFO MS word parity
1	d_left	1 st dword stored (at next dword write qword to ingress fifo)
64	d_data	Stored dword
2	d_par	Stored dword parity
80		Total bits

FIG. 13A

21/38

Slow-Path Reason codes (*=set by dispatcher):

- 0 : nop -- fastpath iSCSI
- 1: ARP frame
- 2: other non-IP (not ARP) frame
- 3: IP fragment (if not, fragment zero could be iSCSI)
- 4: TCP but not iSCSI or runt iSCSI (flen<0x38)
- 5: UDP frame
- 6: ICMP frame
- 7: other IP frame (not IP fragment, TCP, UDP, or ICMP)
- 8: iSCSI, IP fragment zero
- 9: iSCSI, no socket ID found in CAM
- a: iSCSI, unsupported option
- *b: iSCSI, fastpath disabled
- *c: iSCSI, out of sequence
- *d: iSCSI, bad data boundary

FIG. 13B

CAM_LOAD I, data = Write CAM entry I with specified; Set Valid bit;
CAM_READ I = Read data contained in CAM entry I;
CAM_INV I = Clear the valid bit for CAM entry I;
CAM_REQ P = Initiate CAM search with Key elements written in the
CAM-Key register for network port P;
CAM_RESULT P = Fetch result from CAM search for network port P;

FIG. 14

22/38

#bits	Field	Description
2	TS offset	If nonzero, byte offset minus 1 from start to TCP options to timestamp field
4	TCP option length	Size in words of TCP options
4	slow-path Reason Code	<p>If nonzero, packet takes slow path.</p> <p>Reason codes (*=set by dispatcher):</p> <ul style="list-style-type: none"> 0 : nop -- fastpath iSCSI 1: ARP frame 2: other non-IP (not ARP) frame 3: IP fragment (if not, fragment zero could be iSCSI) 4: TCP but not iSCSI or runt iSCSI (flen<0x38) 5: UDP frame 6: ICMP frame 7: other IP frame (not IP fragment, TCP, UDP, or ICMP) 8: iSCSI, IP fragment zero 9: iSCSI, no socket ID found in CAM a: iSCSI, unsupported option *b: iSCSI, fastpath disabled *c: iSCSI, out of sequence *d: iSCSI, bad data boundary
1	ID valid	iSCSI socket ID valid
1	Init	Initiator mode
4	IP option length	Size in words of IP options
10	Socket ID	iSCSI socket ID
1	VLAN	16 th byte contains Vlan tag (IP frame)
1	802.3	802.3 rfc 1042 coding was removed from ethernet header (IP frame)
14	frame length	Length of formatted frame in bytes
16	partial checksum	Checksum for UDP or partial TCP segment (info PIEman)

FIG. 15

23/38

#bit s	field	description	notes
58	job	iff output	
13	roffst	iff read offset	for random access of iff
13	rcpkt	iff read checkpoint (start of frame)	for calculating iff discard point (add length)
10	fctr	frame qword counter	
32	seq	TCP sequence number	
32	ack	TCP acknowledgement number	
32	flgs	flags, TCP flags, TCP window size	if iSCSI (except flags) for msgRxNotify and msgRxFrame, also need some job fields
32	ts	TCP timestamp	
32	ets	TCP echo timestamp	
288	rcp0-rcp8	Rx chunk pointer 0 up to 8	depending on length for msgRxFrame
546		total (approx 69 bytes per port, total 552B)	

FIG. 16

24/38

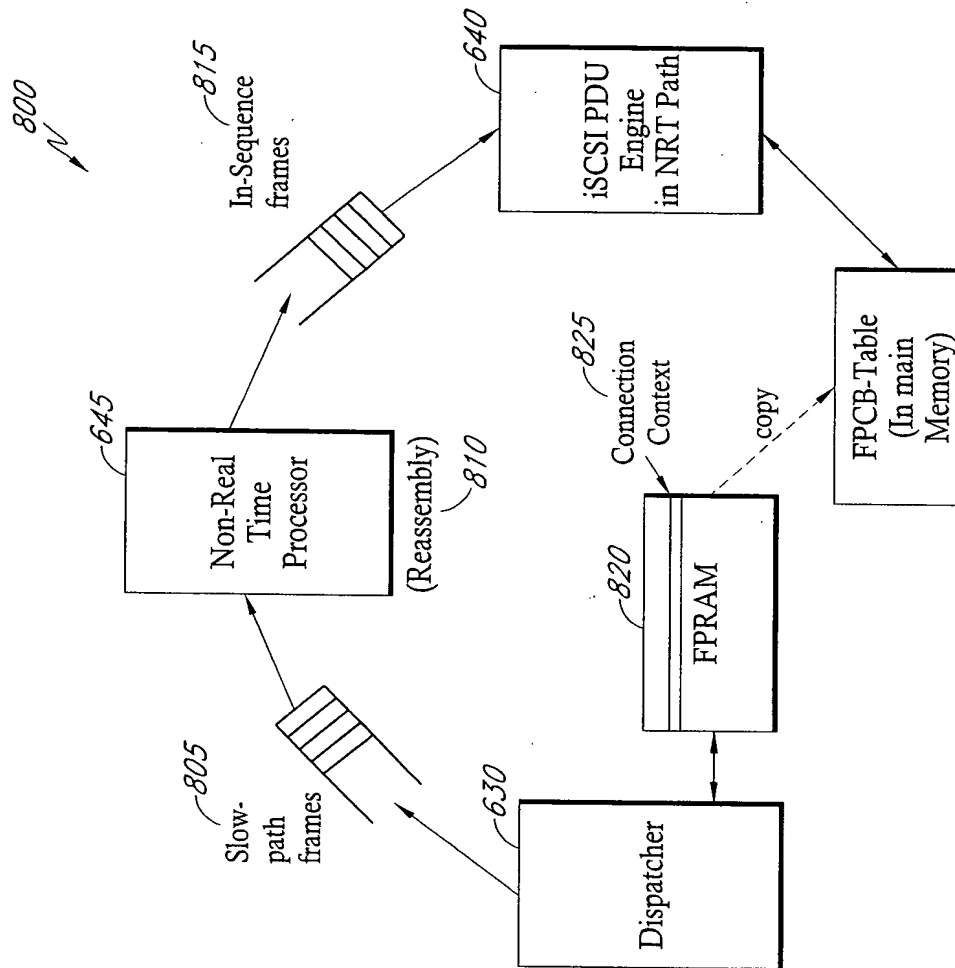


FIG. 17

25/38

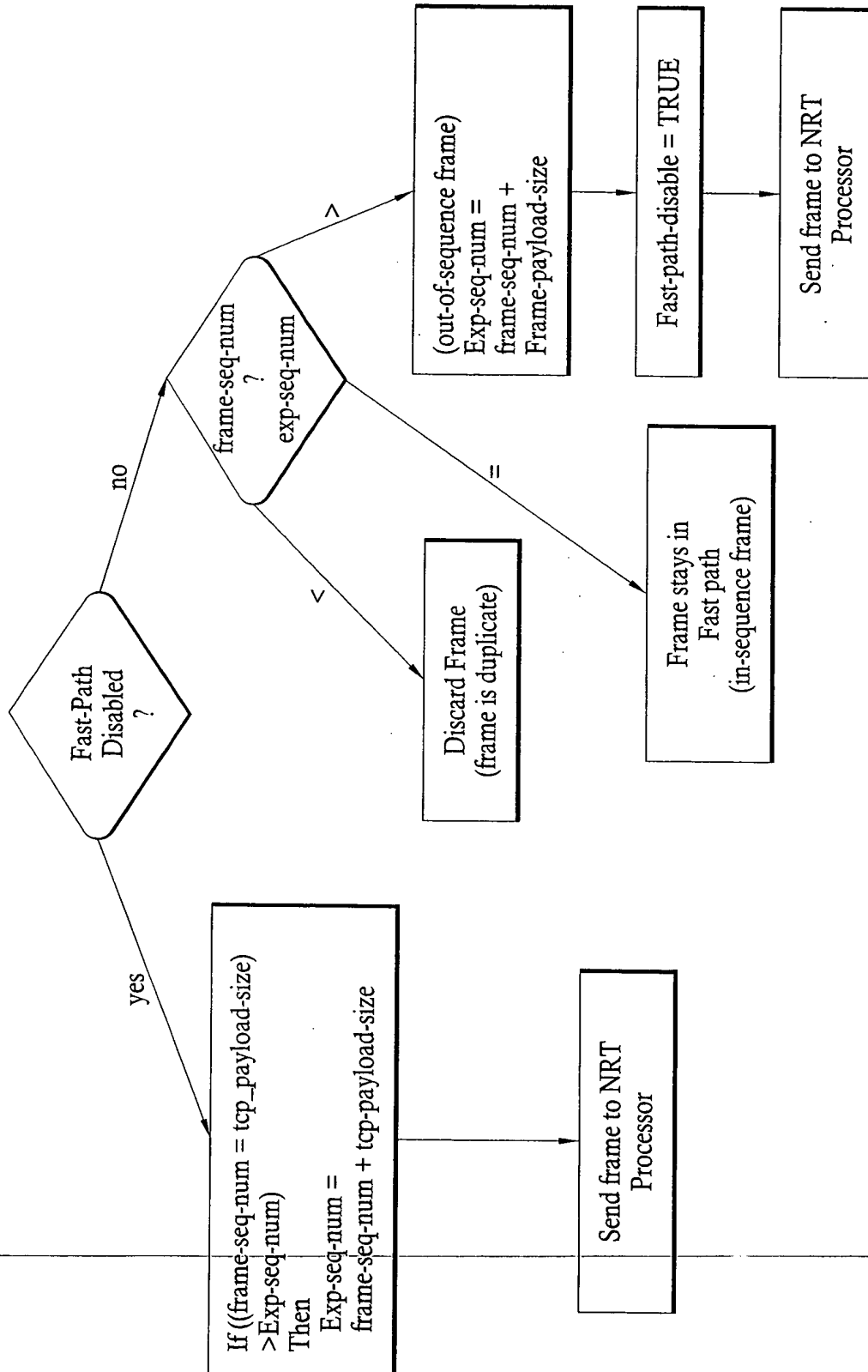


FIG. 18A

byte#	#bits	field
0	8	Control bits: 7: slow path forever (disallows automatic return to fast path) 6: low priority (0=high pri). If set and awr_prx_rdy_lo false, discard frame 5: TCP timestamp valid. If invalid, timestamp check automatically passes 4: PDU Rx disable. Do not store subsequent data or header segments 3: PDU flush. Do not store data segment, auto-reset flush at PDU end 2: data digest enable. Enables check of iSCSI data segment CRC 1: header digest enable. Enables check of iSCSI header segment CRC 0: fastpath enable. If disabled, entire frame stored to Rx chunk.
1-4	32	Next TCP sequence number expected
5	8	TCP timestamp [17:10]
6-9	32	iSCSI partial digest (checked so far)
10-12	24 (1) (23)	iSCSI ata segment control: 23: write SCSI data to buffer memory, vs header/non-SCSI data to Rx queue (flags how to interpret bytes 17-31) 22-0: data segment down counter (# words remaining in data seg, including data digest if present)
13-16	32	SCSI data write pointer ([3:0] indicates #qword residual bytes)
17-31	120 (8) (8) (24) (8) (32) (16) (24) 120	Within iSCSI header segments: [7-4]=spare, [3]=final PDU, [2]=scsi data, [1:0]=#residual bytes [7-5]=spare, [4-0]= state Word residual (up to 3 bytes) Additional header segment down counter (#words in AHSs) Data offset from data out or data in header Data length from DPT Data offset from DPT, bits 31:24 (sb 512B boundary) Within SCSI data segments: Qword residual (up to 15 bytes)

FIG. 18B

27/38

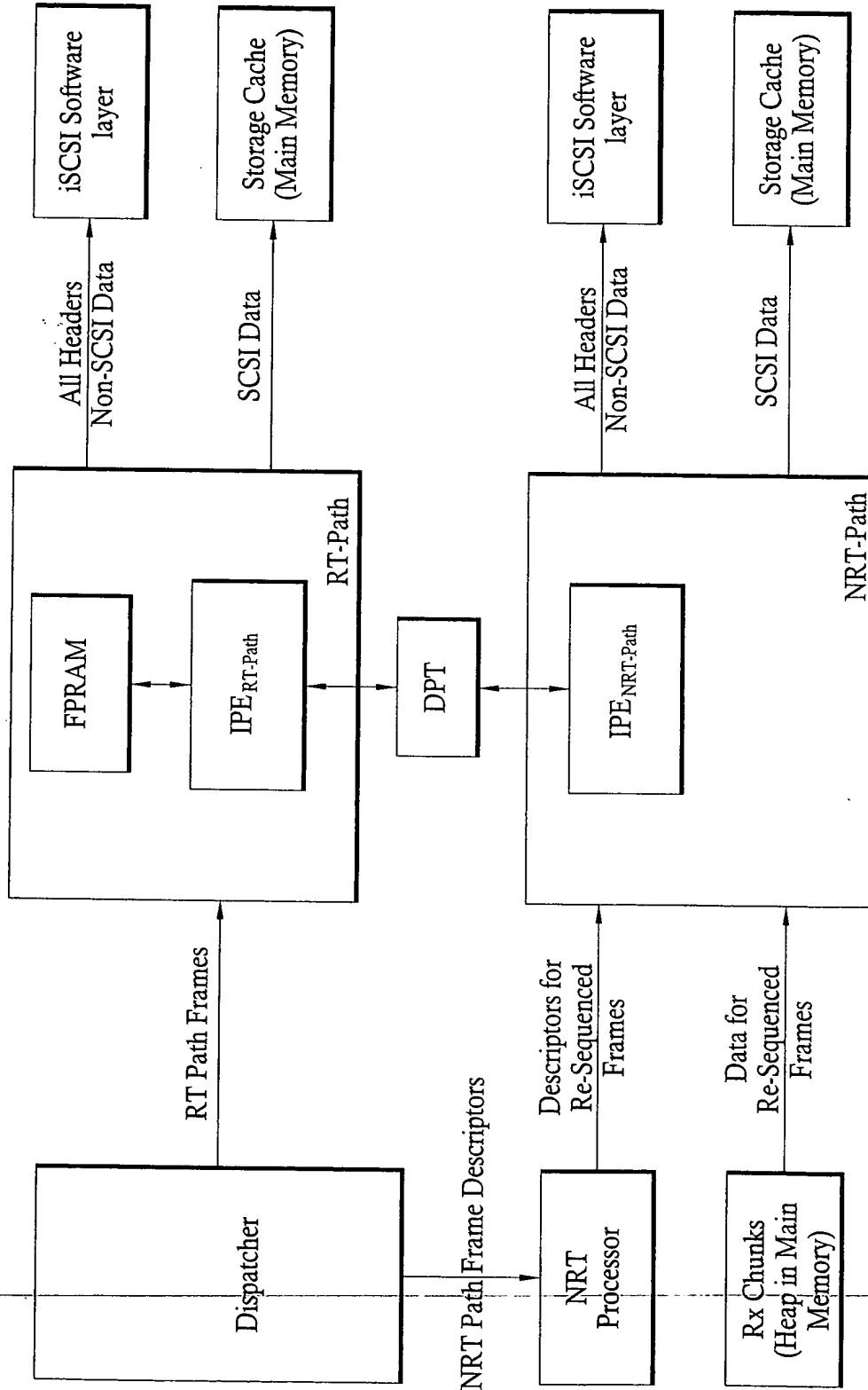


FIG. 18C

28/38

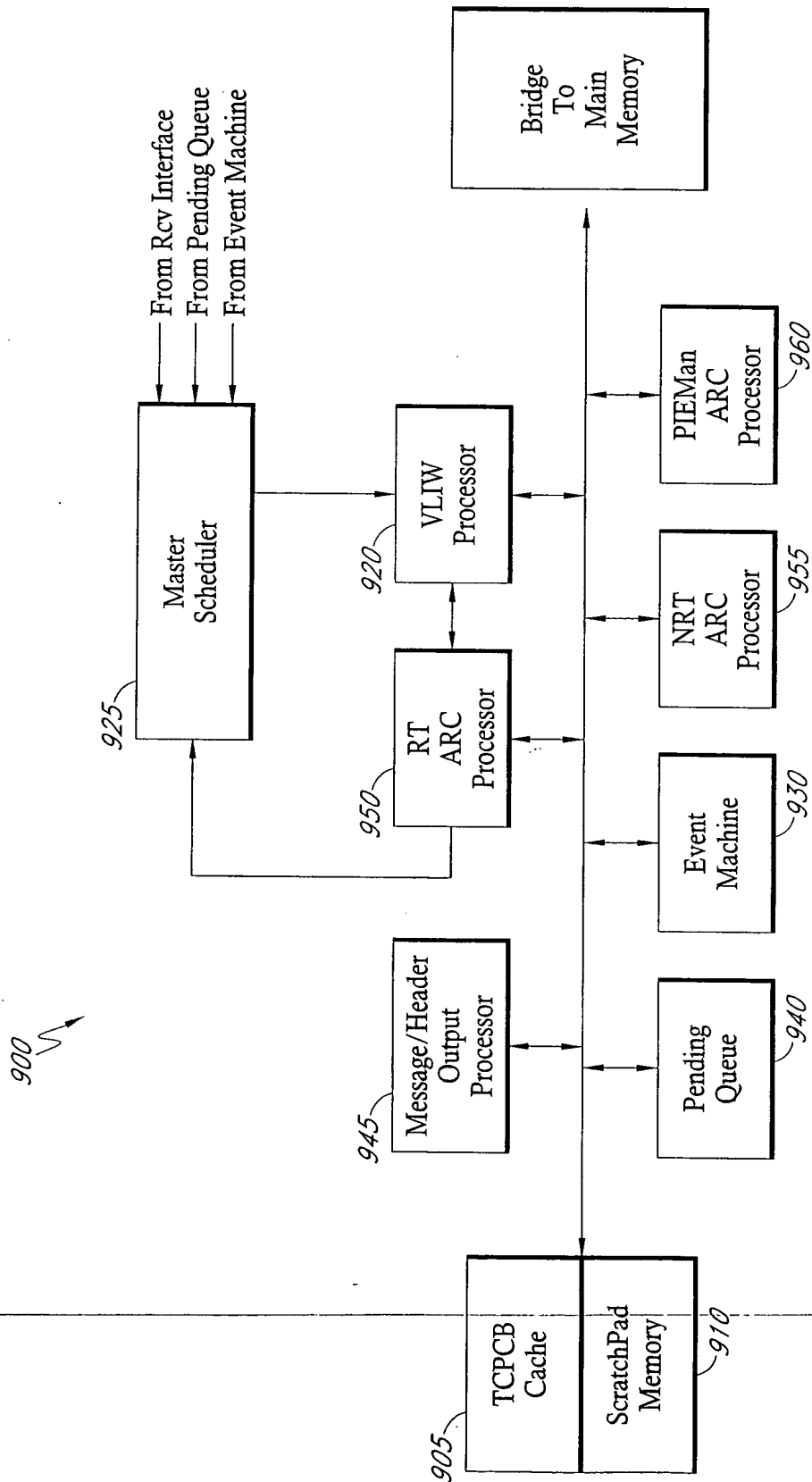


FIG. 19

29/38

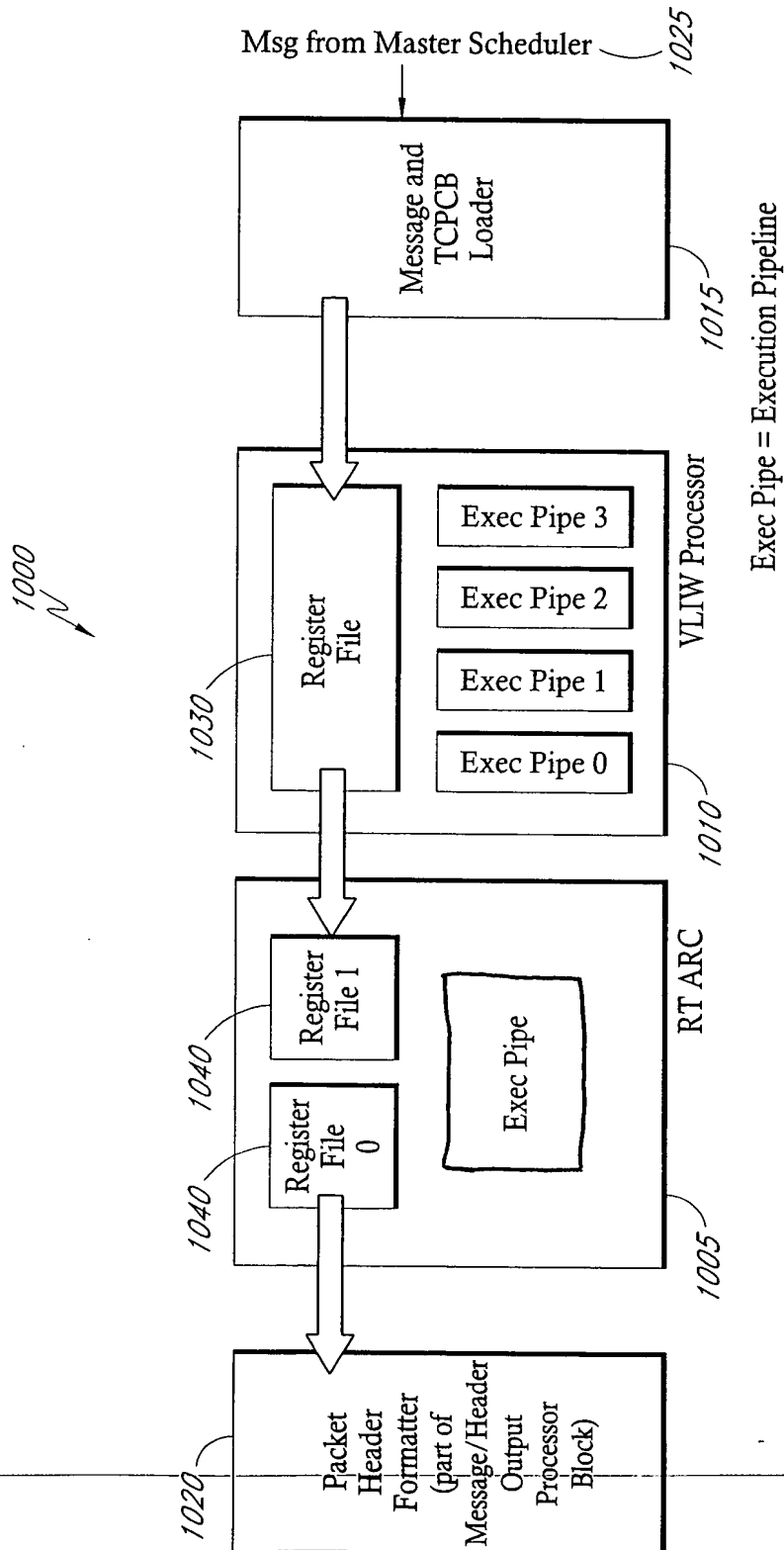


FIG. 20

30/38

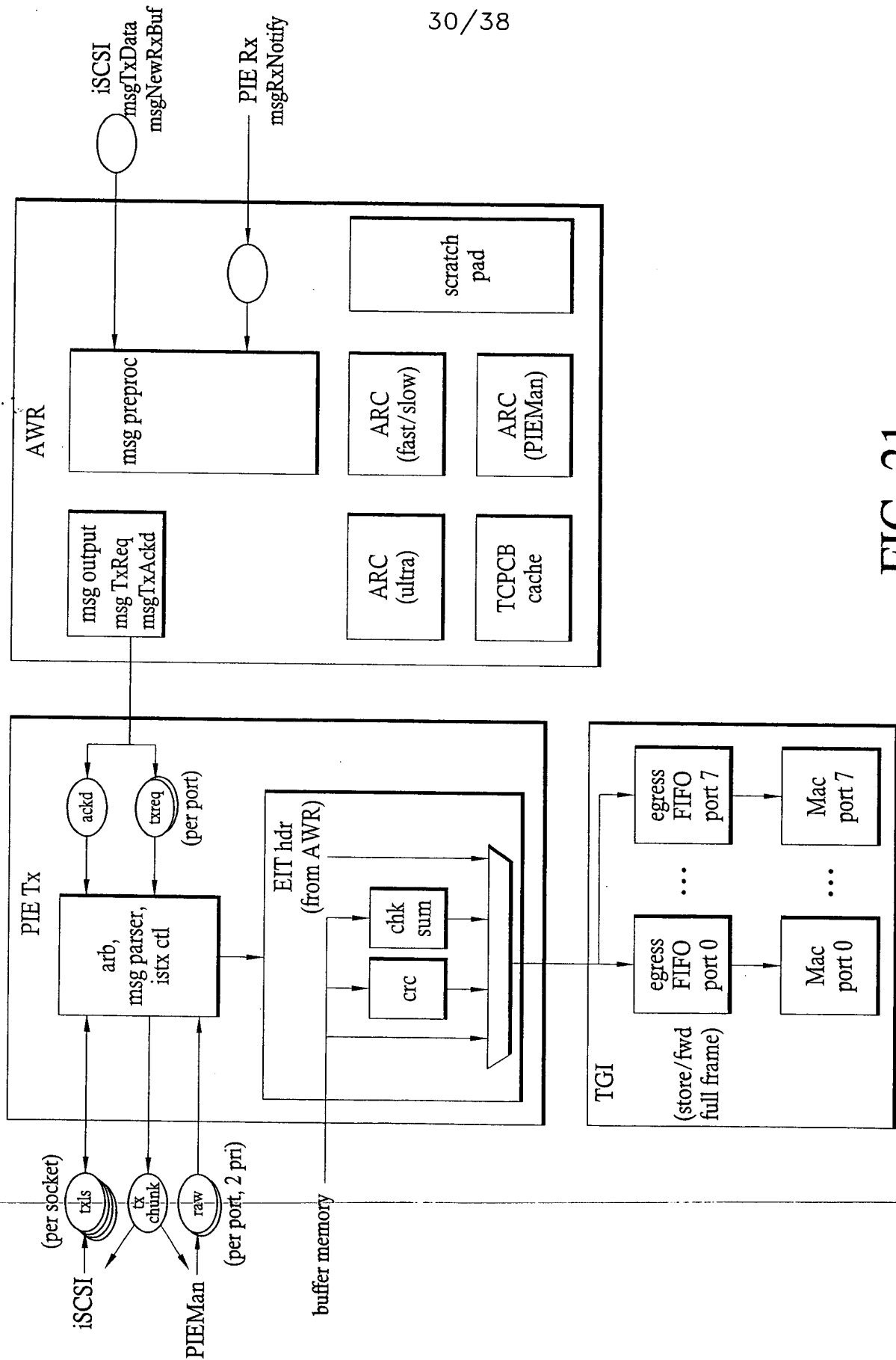


FIG. 21

31/38

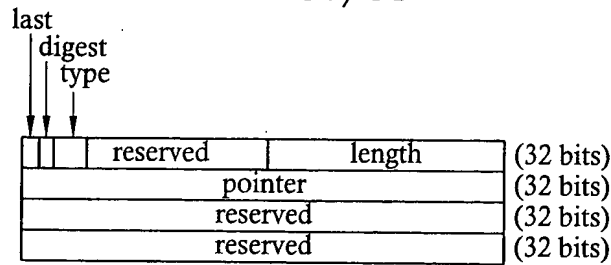


FIG. 22

data in PDU

0	1	h	rsv	48	ptr to hdr in Tx ch	reserved	reserved
0	1	d	rsv	#bytes 1	ptr to data1	reserved	reserved
0	1	d	rsv	#bytes 2	ptr to data2	reserved	reserved
1	1	d	rsv	#bytes 3	ptr to data3	reserved	reserved

login response PDU

0	1	h	rsv	48	ptr to hdr in Tx ch	reserved	reserved
0	1	t	rsv	#bytes	ptr to text	reserved	reserved

FIG. 23

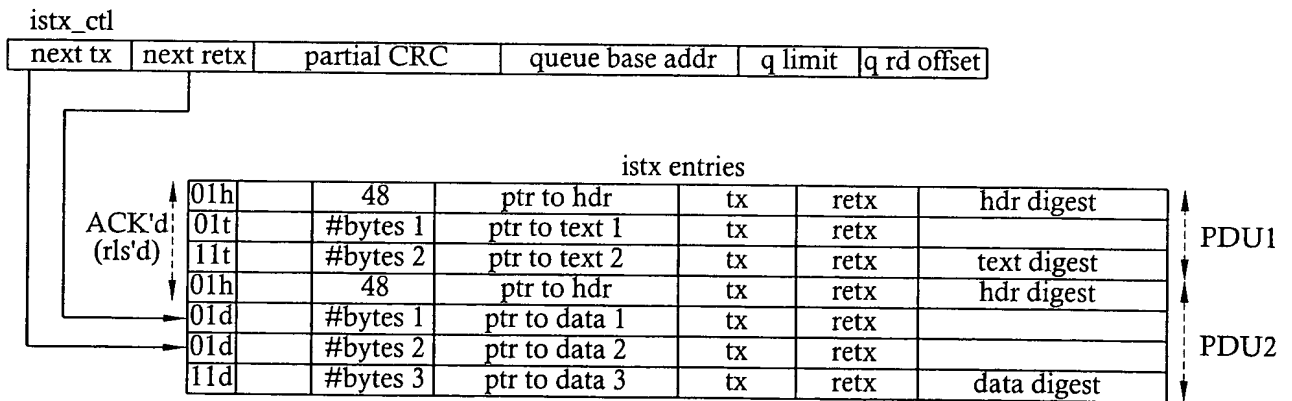


FIG. 24

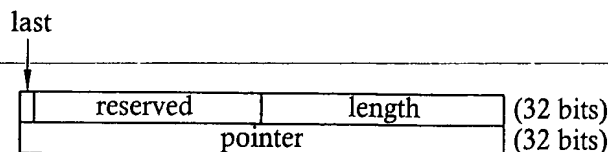


FIG. 25

Command	Description
Push	Writes data beginning at write pointer and saves new write pointer in descriptor.
Push/Inc	Writes data beginning at write pointer, increments counter field by one, and saves new write pointer and counter in descriptor
Inc	Increments counter field by a specified amount and saves new counter in descriptor
Inc Bytes	Increments write pointer field by a special amount and saves new write pointer in descriptor
Push/Chkpt	Writes data beginning at write pointer and saves new write pointer in descriptor and in descriptor extension as write checkpoint.
Push/Inc/Chkpt	Writes data beginning at write pointer, increments counter field by one, saves new write pointer and counter in descriptor, and saves new write pointer in descriptor extension as write checkpoint.
Inc/Chkpt	Increments counter field by a specified amount, saves new counter in descriptor, and copies current write pointer to write checkpoint
Rewind	Copies write checkpoint to write pointer and saves result in descriptor
Peek	Reads data beginning at read pointer (for queues) or write pointer (for stacks) but does not save new pointer in descriptor
Pop	Reads data beginning at read pointer (for queues) or write pointer (for stacks) but saves new pointer in descriptor
Pop/Dec	Reads data beginning at read pointer (for queues) or write pointer (for stacks) decrements counter field by one, and saves new pointer in descriptor
Dec	Decrement counter field by a specified amount and saves new counter in descriptor
Dec Bytes	Decrement read pointer (for queues) or write pointer (for stacks) by a specified amount and saves new pointer in descriptor

FIG. 26

33/38

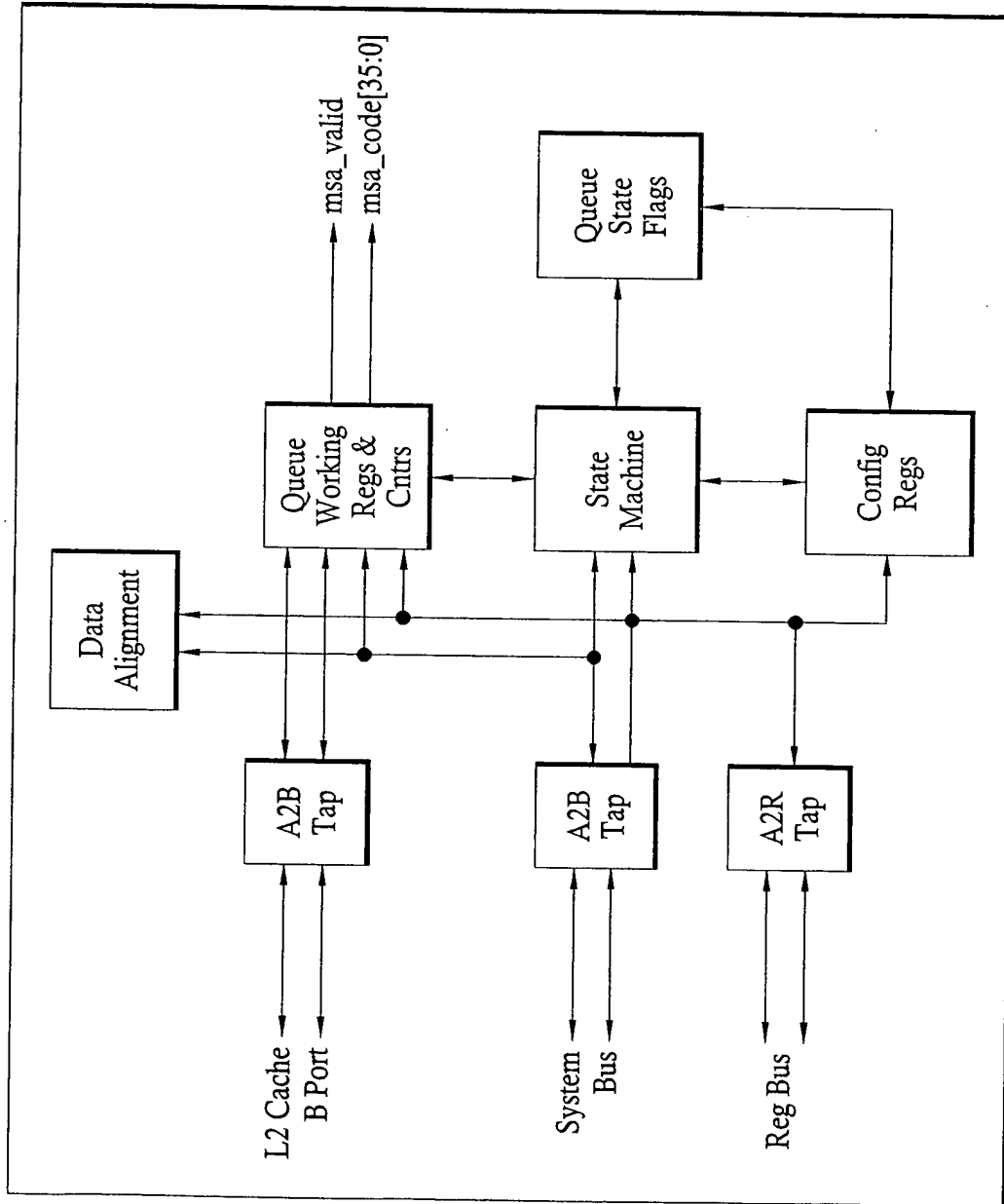


FIG. 27

34/38

00000	---	01000	---	10000	---	11000	Rewind
00001	Pop/Dec/ (Read)	01001	Pop	10001		11001	Peek
00010	Push/Inc (Write)	01010	Push	10010	Push/Inc/Chkpt	11010	Push/Chkpt
00011	---	01011	Dec	10011	---	11011	---
00100	---	01100	---	10100	---	11100	---
00101	---	01101	---	10101	---	11101	---
00110	---	01110	Inc	10110	Inc/Chkpt	11110	---
00111	---	01111	---	10111	---	11111	---

FIG. 28

00	Queue Not Empty	10	Queue Underflow
01	Queue Empty	11	Not used

FIG. 29

35/38

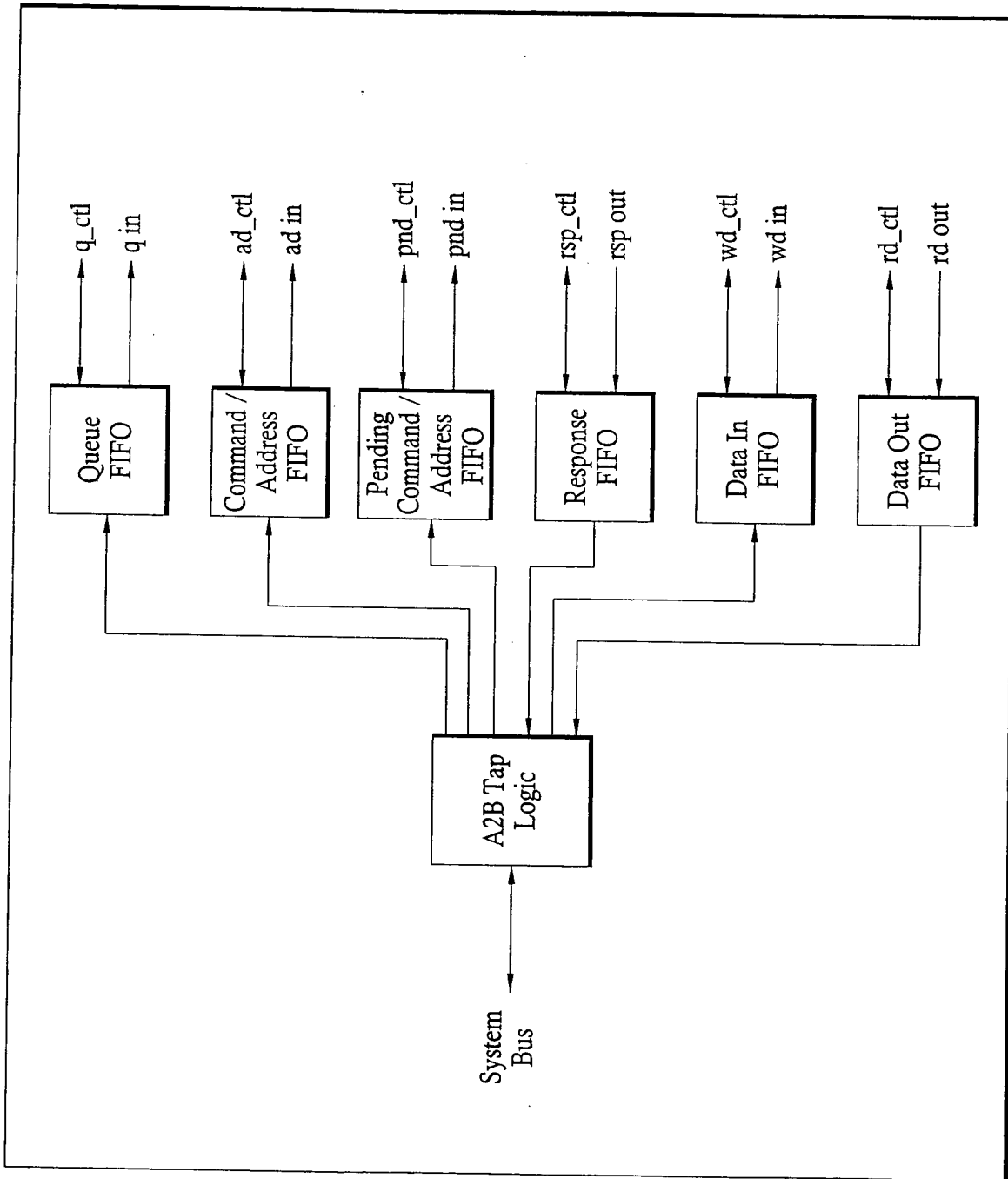


FIG. 30

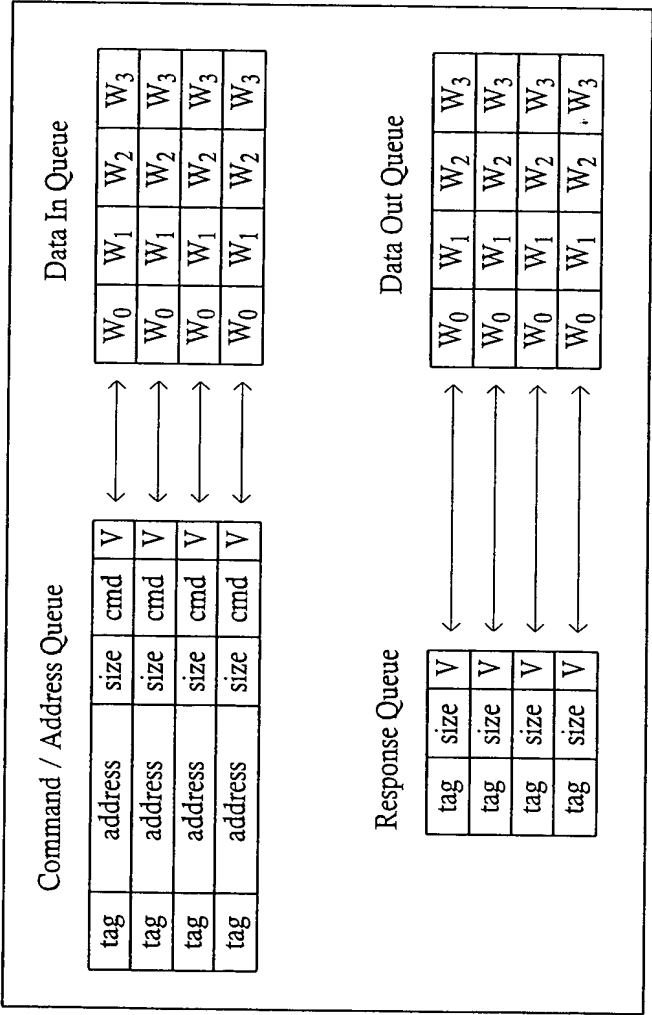
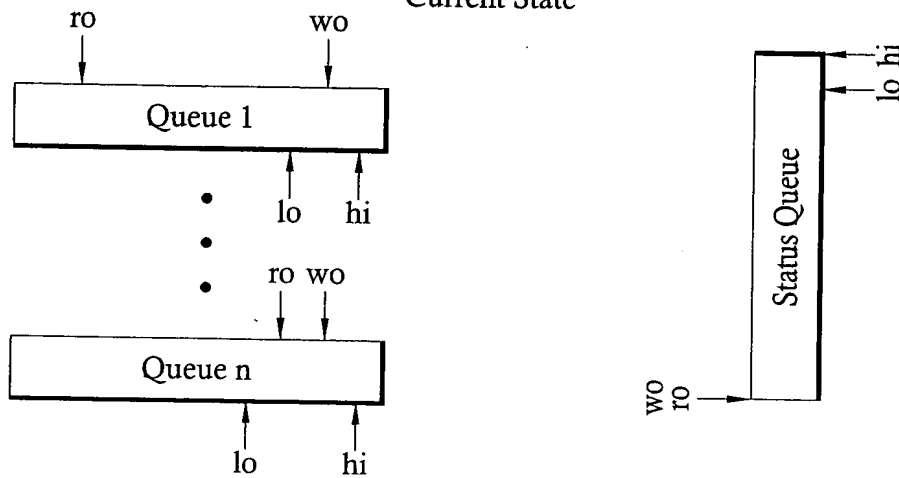


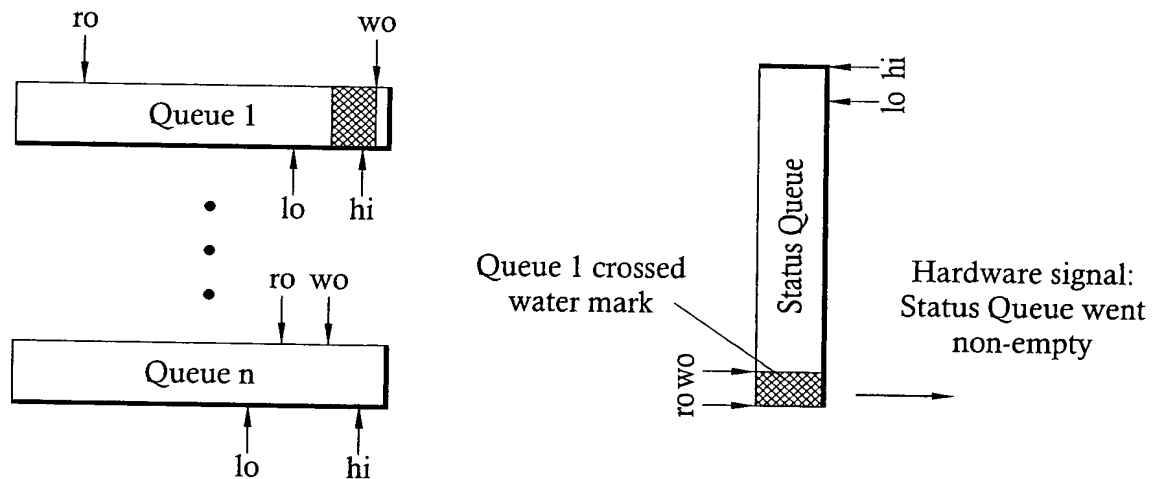
FIG. 31

37/38

Current State



Push to Queue 1



Pop from Queue n

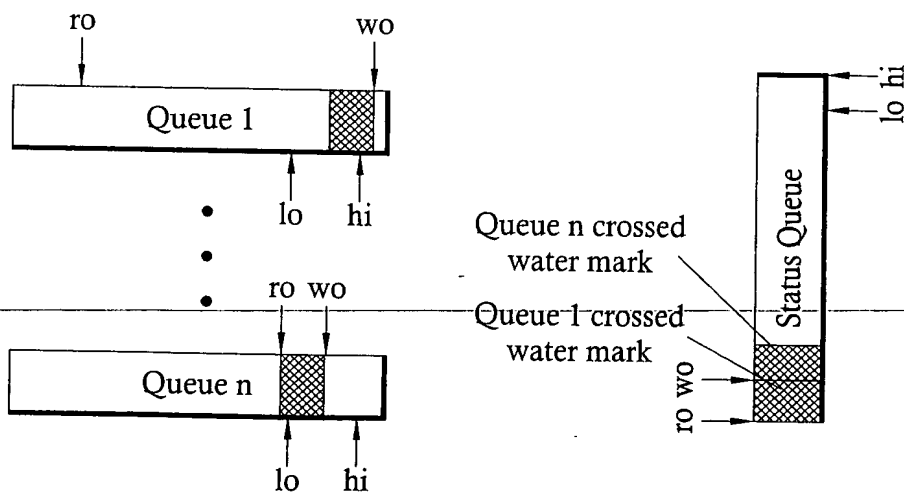


FIG. 32

38/38

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Full Hi Level	Full Lo Level	Read Pointer = 0	Write Pointer = 0	Count = 0	Not Empty Signal	Base Address of Data	M Size		0						

FIG. 33

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Full Hi Level	Full Lo Level	Read Pointer = 0	Write Pointer = bytes in system memory	Count = number of units in system memory	Not Empty Signal	Base Address of Data = base address of data written to system memory	M Size		0						

FIG. 34